SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

SREE SAINATH NAGAR, A. RANGAMPET - 517 102.

(Affiliated to Jawaharlal Nehru Technological University Anantapur, Ananthapuramu)

ASSIGNMENT

P.C.K. Chap.

Name of the Student :M.NIKHIL KUMAR REDDY H.T.No. 1 9 1 2 1 A 0 5 D 9

Year & Semester : II YEAR & II SEMESTER Branch : CSE Section : B

Name of the Subject :DESIGN & ANALYSIS OF ALGORITHMS Academic Year: 2020-2021

Signature of the Student : M.NIKHIL Signature of the Faculty:

Q.No.	1	2	3	4	5
Marks Awarded					

Total Marks

VII 1. what is external Sorting? Explain any one external Sorting algorithm with an example

External Sorting is a class of Sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of computing device (usually RAM) and instead they must reside in the slower external memory, usually a hard disk drive. Thus, external sorting algorithms are external memory algorithms and thus applicable in the external memory model of computation.

External Sorting algorithms generally fall into two types, distribution sorting, which resembles quick sort, and external merge sort, which resembles merge sort. The latter typically uses a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in main memory are read, sorted, and written out to a temporary file. In the merge phase, the sorted subfiles are combined into a single larger file.

External merge Sort:

One example of external sorting is the external merge sort algorithm, which is a K-way merge algorithm. It sorts chunks that each fit in RAM, then merges the sorted chunks together.

The algorithm first sorts M items at a time and puts the sorted lists back into external memory. It then recursively does a $\frac{m}{B}$-way merge on those sorted lists. To do this merge, B elements from each sorted list are loaded into internal memory, and the minimum is repeatedly outputted.

For example, for sorting 900 megabytes of data using only 100 megabytes of RAM:

1. Read 100 MB of the data in main memory and sort by some conventional method, like quicksort.
2. Write the sorted data to disk.
3. Repeat steps 1 and 2 until all the data is in sorted 100 MB chunks (there are 900 MB / 100 MB = 9 chunks), which now need to be merged into one single output file.
4. Read the first 10 MB (= 100 MB / (9 chunks + 1)) of each sorted chunk into input buffers in main memory and allocate the remaining 10 MB for an output buffer. (In practice, it might provide better performance to make the output buffer larger and the input buffers slightly smaller.)

5. Perform a 9-way merge and store the result in the output buffer. whenever the output buffer fills, write it to the final sorted file and empty it. whenever any of the 9-input buffers empties, fill it with the next 10 MB of its associated 100 MB sorted chunk until no more data from the chunk is available. This is the key step that makes external merge sort work externally - because the merge algorithm only makes one pass sequentially through each of the chunks, each chunk does not have to be loaded completely; rather, sequential parts of the chunk can be loaded as needed.

Historically, instead of a sort, sometimes a replacement-selection algorithm was used to perform the initial distribution, to produce on average half as many output chunks of double the length.

2. Explain Splay trees algorithm with an example. Analyse its time complexity.

A splay tree is a self-balancing tree or self-adjusted binary search trees. In other words, we can say that the splay trees are the variants of the binary search trees. A splay tree is a self-balancing tree, but AVL and Red-Black trees are also self balancing trees. It has one extra property that makes it unique is splaying.

A splay tree contains the same operations as a Binary Search tree, i.e. Insertion, deletion and searching. but it also contains one more operation i.e. splaying. So, all the operations in the splay tree are followed by splaying.

Splay trees are not strictly balanced trees, but they are roughly balanced trees. The rearrangement of the tree will be done through the rotations.

There are six types of rotations used for splaying:

1. Zig rotation (Right rotation)
2. Zag rotation (Left rotation)
3. Zig Zag (Zig followed by zag)
4. Zag Zig (Zag followed by Zig)
5. Zig Zig (two right rotations)
6. Zag Zag (two left rotations)

Cases for the Rotations

Case 1: If the node does not have a grand parent and if it is the right child of the parent, then we carry out the left rotation; otherwise, the right rotation is performed.

Case 2: If the node has a grandparent, then based on the following Scenarios; the rotation is performed.

Scenario 1: If the node is the right of the parent and the parent is also right of its parent, then zig zig right right operation is performed.

Scenario 2: If it is at the left, then zig zag right left rotation is performed.

Scenario 3: If the node is right of the parent and the pasent is right of its pasent, then zig zig left left rotation is performed.

Scenario 4: If it is at the left, then zig zag right-left rotation is performed.

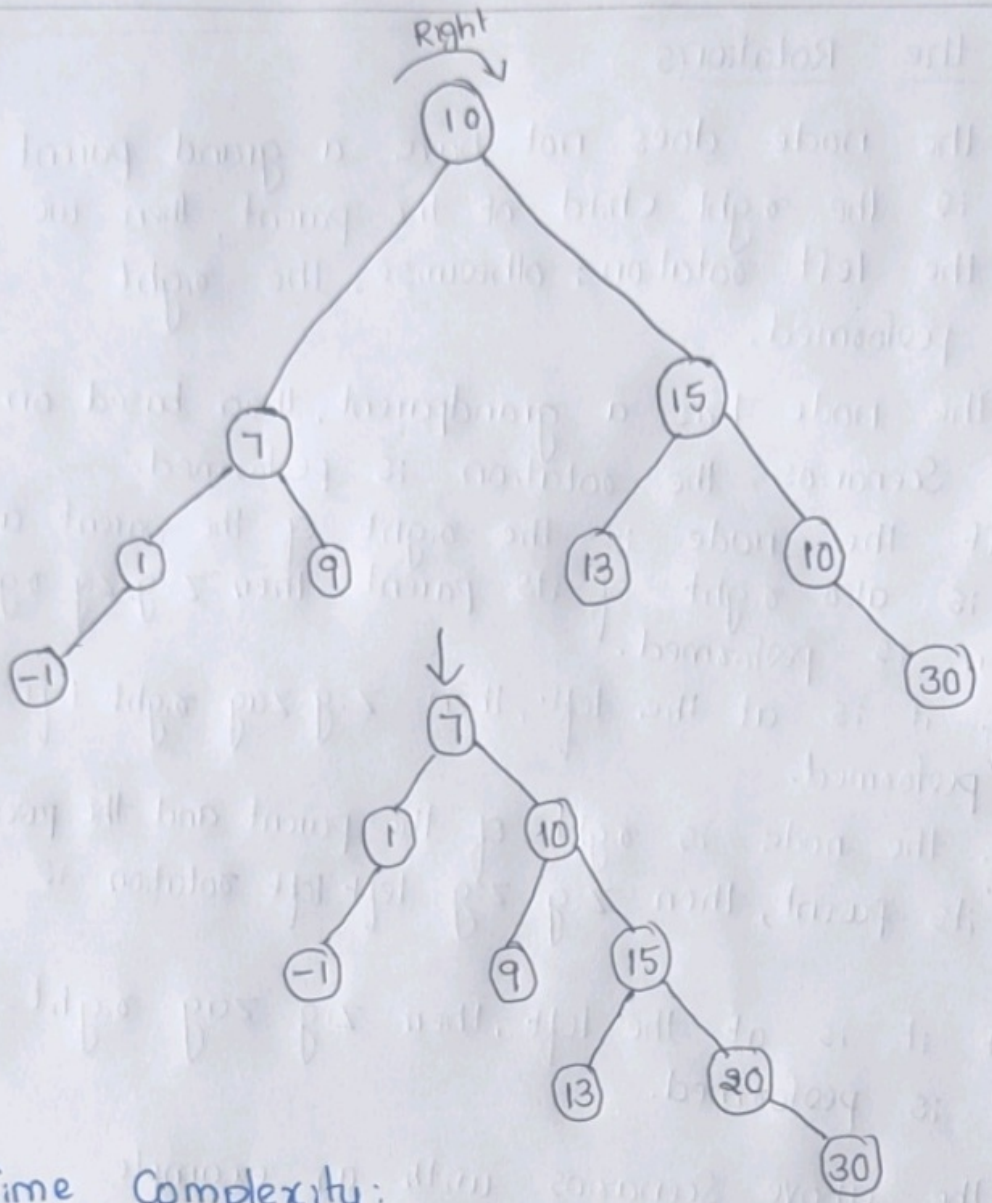
Let's look the above Scenarios with an example.

consider the below example:

For example, we have to search 7 element in the tree. we will follow the below steps:

Step 1: First we compare 7 with a root node. As 7 is less than 10, so it is a left child of the root node.

Step 2: once the element is found, we will perform splaying. The right rotation is performed so that 7 becomes the root node of the tree



Time Complexity:

All Splay tree operations run in O(log n) time on average where n is the number of entries in the tree. Any single operation can take Theta(n) time in the worst case. The Search operation in Splay tree does the standard BST Search, in addition to Search, it also splays (move a node to the root).

Time Complexity in big O notation:

Algorithm	Average	worst case
Space	O(n)	O(n)
Search	amortized O(log n)	amortized O(log n)
Insert	amortized O(log n)	amortized O(log n)
Delete	amortized O(log n)	amortized O(log n)